BÜSCHER, Carina
Köln

"Vielleicht 360° geteilt durch die Anzahl der Winkel?" – Wege zur Bestimmung des Drehwinkels beim Programmieren regelmäßiger n-Ecke

Unser Alltag ist zunehmend von digitalen Systemen geprägt. Menschen sollten in dieser digitalisierten Welt nicht nur passive Nutzer, sondern aktive Gestalter sein (Gadeib, 2019). In dem Zusammenhang wird Computational Thinking als eine wichtige Fertigkeit im 21. Jahrhundert angesehen (Wing, 2006), weshalb bereits in der Schule entsprechende Kompetenzen aufgebaut werden sollten. Wing (2006) nennt sie in einer Reihe mit Lesen, Schreiben und Rechnen. Computational Thinking beschreibt eine Denkweise, die es erlaubt ein Problem auf eine Art und Weise zu beschreiben, dass ein Computer es lösen könnte (ebd.).

Auch wenn Computational Thinking mehr ist als Programmieren, so können eigene Erfahrungen mit Programmierung das Computational Thinking der Lernenden fördern. Programmieren stellt aber nicht nur einen wichtigen Lerngegenstand dar, sondern eröffnet auch neue Perspektiven auf mathematische Inhalte, wodurch ein vertieftes Verständnis dieser gefördert werden kann. Papert hat bereits 1980 in der von ihm entwickelten Turtle-Geometrie eine andere Art gesehen, Geometrie zu betreiben: “Turtle geometry is a different style of doing geometry, just as Euclid’s axiomatic style and Descartes’s analytic style are different from one another. Euclid’s is a logical style. Descartes’s is an algebraic style. Turtle geometry is a computational style of geometry.” (Papert, 1980, S. 69).

Computational Thinking spielt zunehmend auch im Mathematikunterricht eine Rolle. Einige Studien beschäftigen sich beispielsweise mit der Frage, wie CT als Lerngegenstand auch im Mathematikunterricht thematisiert werden kann (Weintrop et al., 2016). Andere Studien untersuchen, inwiefern CT bzw. Programmieraktivitäten förderlich für das Mathematiklernen sein können (Benton et al., 2017). Harlow et al. (2016) zeigen erste Einblicke, wie Kinder beim Entwerfen von Algorithmen für Turtle-Grafiken sowohl mathematische Konzepte selbstständig entwickeln, als auch algorithmische Konzepte wie Schleifen. Dies zeigt das Potenzial, Programmierkonzepte und mathematische Konzepte integriert zu fördern. Benton et al. (2017) haben im Rahmen des Entwicklungsforschungsprojekts “ScratchMath”, Materialien für Lernende im Alter von 9 bis 11 und Fortbildungsmaterialien für Lehrkräfte entwickelt. Die Lernenden wurden u.a. dazu angeregt, den Zusam-

In: P. Ebers, F. Rösken, B. Barzel, A. Büchter, F. Schacht & P. Scherer (Hrsg.),
Beiträge zum Mathematikunterricht 2024.
57. Jahrestagung der Gesellschaft für Didaktik der Mathematik. WTM.
https://doi.org/10.37626/GA9783959872782.0

menhang zwischen der Anzahl der Drehungen und der Größe des Drehwinkels zu erkunden. Basierend darauf sollten unter anderem Programmcodes erstellt werden, mit denen verschiedene regelmäßige n-Ecke gezeichnet werden sollten. Insgesamt berichten sie, dass sowohl die Idee des Algorithmus als auch die 360° Drehung sowohl für Lernende als auch für Lehrkräfte eine Herausforderung darstellt. Es besteht die Gefahr, dass Lernende die mathematischen Elemente möglicherweise übersehen (Hoyles & Noss, 1992). Die angelegten Potenziale für das Mathematiklernen werden also nicht zwangsläufig automatisch ausgeschöpft.

In einer ersten Analyse des hier vorgestellten Projekts wurden zunächst die CT-Aktivitäten von Lernenden beim Programmieren regelmäßiger n-Ecke rekonstruiert und dabei empirisch genauer spezifiziert (Büscher, eingereicht). Dabei wurden für die CT-Aktivitäten Algorithmisieren, Testen und Evaluieren, Wiedernutzen und Abändern, Zerlegen, und Abstrahieren jeweils Unterkategorien gebildet, die es ermöglichen, die CT-Aktivitäten von Lernenden genauer zu beschreiben. Nun geht es darum zu rekonstruieren, welche mathematischen Lernprozesse initiiert werden können. In diesem Beitrag steht dazu zunächst die Frage im Fokus, wie Lernende den Drehwinkel beim Programmieren von Turtle-Grafiken bestimmen. Dieser Beitrag widmet sich daher der Forschungsfrage: Wie bestimmen Lernende den Drehwinkel beim Programmieren von Turtle-Grafiken zu regelmäßigen n-Ecken?

Methoden

Die hier vorgestellte Studie ist Teil eines größeren Entwicklungsforschungsprojekts zur integrierten Förderung von CT und mathematischen Vorstellungen im Geometrieunterricht (Büscher, eingereicht). In diesem Beitrag werden die Ergebnisse einer explorativen Fallstudie (Yin, 2002) zur empirischen Rekonstruktion von Wegen zur Bestimmung des Drehwinkels beim Programmieren regelmäßiger n-Ecke präsentiert. Dazu wurden 6*2 Designexperimente mit 12 Lernenden im Alter von 11-13 durchgeführt. Diese wurden videographiert (insgesamt ca. 720 Minuten Videomaterial) und transkribiert. Für die Auswertung wurden zunächst alle Stellen identifiziert, in denen es um Bestimmung des Drehwinkels ging. Diese Szenen wurden einer qualitativen Inhaltsanalyse (Kuckartz & Rädiker, 2022) unterzogen mit Blick auf die geäußerten Ideen und Ansätze zur Bestimmung des Drehwinkels und daraus Kategorien gebildet, um die aufgetretenen Ansätze zu kategorisieren.

Design

Nachdem sie sich mit dem TurtleCoder (www.code-your-life.org/turtlecoder) vertraut machen konnten und Quadrate programmiert haben, war die Aufgabe, mit dem TurtleCoder weitere regelmäßige n-Ecke zu zeichnen.

Beim Entwerfen eines Algorithmus zum Zeichnen ebener Figuren reicht es nicht, die Eigenschaften der zu zeichnenden Figur statisch zu beschreiben (ein Hexagon hat sechs gleich lange Seiten und sechs gleich große Innenwinkel à 120°). Durch das Programmieren müssen diese Eigenschaften dynamisch gedacht werden (Die Schildkröte bewegt sich sechsmal dieselbe Länge vorwärts und dreht sich dabei jeweils um 60°). Der Unterschied ist insbesondere, dass die Betrachtung des Innenwinkels nicht reicht, sondern dass man den Außenwinkel dynamisch als Drehbewegung deuten muss.

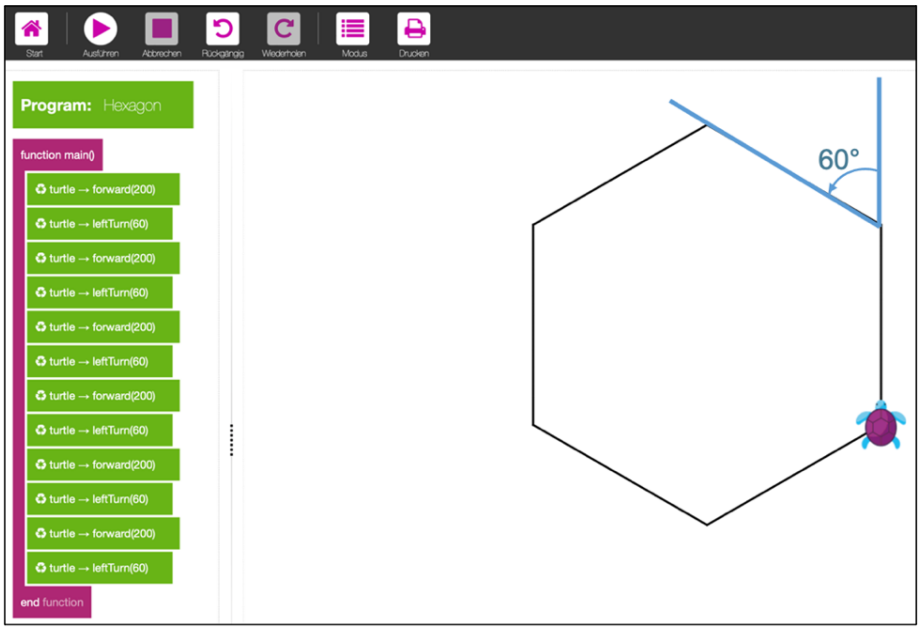

Abbildung 1: Beispielcode zum Zeichnen eines regelmäßigen Sechsecks (erstellt mit dem TurtleCoder unter code-your-life.org). Visualisierung des Außenwinkels für diesen Beitrag.

Ergebnisse

Es konnten verschiedene Wege zur Bestimmung des Drehwinkels beim Programmieren regelmäßiger n-Ecke rekonstruiert werden.

- Nachfragen bei Lehrperson
- unsystematisches Ausprobieren
- systematisches Ausprobieren
- Anfertigen einer Skizze
- Einsetzen des Innenwinkels der gewünschten Figur
- Abstrahieren von der Außenwinkelsumme und gleichmäßiges Verteilen dieser auf die gewünschte Anzahl an Ecken

Diese werden im Vortrag anhand exemplarischer Fallbeispiele erläutert. Aus den rekonstruierten Ansätzen lässt sich ableiten, welche Vorstellungen für ein tragfähiges Bestimmen des Drehwinkels notwendig wären. Somit kann

der Lerngegenstand genauer spezifiziert und das Design zukünftig fokussierter darauf angepasst werden. Erste Beobachtungen dazu deuten an, dass ein explorierendes Testen und Evaluieren zusammen mit einem Wiedernutzen und Abändern der Parameter im Code zu einer besseren Abschätzung des Drehwinkels führen kann. Wenn ein explorierendes Testen und Evaluieren nicht mit einem systematischen Wiedernutzen und Abändern der Parameter des Codes einhergeht, geben Lernende eher auf.

Fazit

Papert (1980) sieht in der Turtle-Geometrie eine weitere Art von Geometrie. Diese nimmt eine dynamische Perspektive ein. Die empirischen Ergebnisse dieser Studie zeigen in diesem Zusammenhang verschiedene Wege zur Bestimmung des Drehwinkels auf. In vielen dieser rekonstruierten Wegen wird auf die ein oder andere Art ausprobiert. Eine zentrale Rolle dabei scheint das automatische Feedback zu spielen. Die Lernenden können beim Ausführen des Codes sofort erkennen, ob die Schildkröte sich auf die gewünschte Art bewegt. Sollte das nicht der Fall sein, können sie anhand der konkreten falschen Bewegung überlegen, wo genau der Fehler liegt und so ihre räumliche Orientierung verbessern. Es deuten sich damit einige Potenziale des Programmierens im Geometrieunterricht an, die es weiter zu untersuchen gilt.

Literatur

Benton, L., Hoyles, C., Kalas, I., Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*(2), 115–138.

Büscher C. (eingereicht). *Differences in students' Computational Thinking activities when designing an algorithm for drawing plane figures*. Eingereichter Journalartikel.

Gadeib, A. (2019). *Die Zukunft ist menschlich – Manifest für einen intelligenten Umgang mit dem digitalen Wandel in der Gesellschaft*. Gabal.

Harlow, D. B., Dwyer, H., Hansen, A. K., Hill, C., Iveland, A., Leak, A. E., & Franklin, D. M. (2016). Computer programming in elementary and middle school: Connections across content. In M. J. Urban & David A. Falvo (Hrsg.), *Improving K-12 STEM education outcomes through technological integration* (S. 337–361). IGI Global.

Hoyles, C., & Noss, R. (1992). *Learning mathematics and logo*. MIT Press.

Kuckartz, U. & Rädiker, S. (2022). *Qualitative Inhaltsanalyse: Methoden, Praxis, Computerunterstützung: Grundlagentexte Methoden*. Beltz.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yin, R. K. (2002). *Case study research*. SAGE.